Jun.-Prof. Dr. Bernhard Beckert
Ulrich Koch

UNIVERSITÄT KOBLENZ · LANDAU

Einführung in die Theoretische Informatik I/ Grundlagen der Theoretischen Informatik
Sommersemester 2007
10. Aufgabenblatt

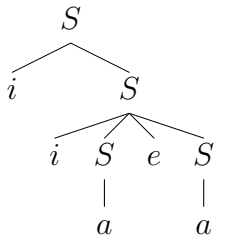
Ausgabe: 25. 06. 2007 Besprechung: 03./04. 07. 2007

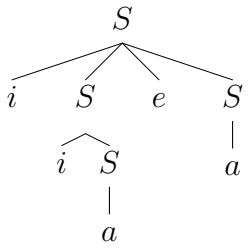
---

1 Mehrdeutige kontextfreie Grammatiken

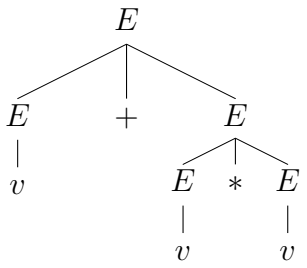
Lösung:

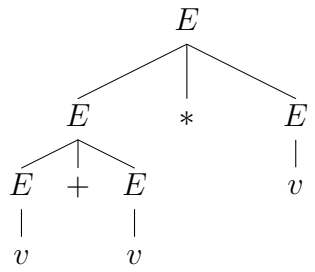
1. Für G_1: $iiaea$



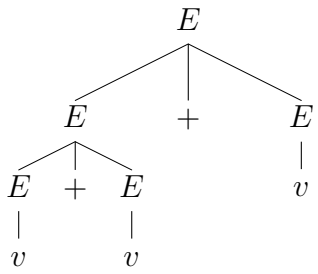


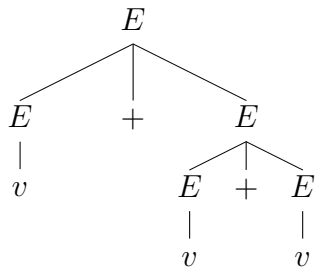
Für G_2: $v + v * v$





Auch für G_2: $v + v + v$





2. G_1 zeigt das so genannte *dangling else problem*: Zu welchem `if` gehört ein `else`? Folgendes Java-Fragment entspricht *iiaea*:

```
if(b1)
  if(b2)
    x = y;
  else
    y = x;
```

Wie die Einrückung suggeriert, gehört in Java (und C, C++, Pascal, Delphi, ...) ein `else` jeweils zum letzten `if`, das „noch kein `else` hat". Folglich bildet der zuerst angegebene Ableitungsbaum die Java-Semantik besser ab.

G_2 erzeugt arithmetische Ausdrücke, allerdings ohne die in Java (und fast allen anderen Programmiersprachen) vorgegebenen Präzedenzen und Assoziativitäten abzubilden: $*$ soll enger binden als $+$, und beide Operatoren sollen linksassoziativ sein (z.B. soll $v + v + v$ dasselbe bedeuten wie $(v + v) + v$). Auch hier ist also jeweils der erste Ableitungsbaum besser.

3. $G'_1 = (\{S, M, U\}, \{i, e, a\}, R'_1, S)$ mit folgenden Regeln in R'_1:

$$\begin{array}{rcl} S & \rightarrow & M \mid U \\ M & \rightarrow & iMeM \mid a \\ U & \rightarrow & iS \mid iMeU \end{array}$$

Hier steht M für *matched statement* und U für *unmatched statement*. Dabei ist s ein *matched statement* gdw. es keine Anweisung s' gibt, so dass ses' eine Anweisung ist. (Anders gesagt: Ein *matched statement* „kann keinen `else`-Zweig (mehr) bekommen".) s ist ein *unmatched statement* gdw. für jede Anweisung s' gilt, dass ses' eine Anweisung ist. (Also kann ein *unmatched statement* „noch einen `else`-Zweig bekommen".)

$G'_2 = (\{E, T, F\}, \{+, *, v\}, R'_2, E)$ mit folgenden Regeln in R'_2:

$$\begin{array}{rcl} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T * F \mid F \\ F & \rightarrow & v \end{array}$$

T steht für *term*, F für *factor* — diese Bezeichnungen tauchen in manchen Grammatiken für Programmiersprachen auf.

2 Turing-Maschinen

Lösung:

$(\{s_0, s_1, s_a, s_b, s'_a, s'_b, s_Y, s_N, s_f\}, \{a, b, 0, 1, \#\}, \delta, s_0)$ mit:

$$
\begin{array}{rcll}
\delta(s_0, \#) & = & (s_1, L) & \text{zum letzten Zeichen gehen} \\
 & & & \\
\delta(s_1, a) & = & (s_a, L) & a \text{ speichern, erstes Zeichen suchen} \\
\delta(s_1, b) & = & (s_b, L) & b \text{ speichern, erstes Zeichen suchen} \\
\delta(s_1, \#) & = & (s_Y, R) & \text{auf Antwort 1 für } \varepsilon \text{ vorbereiten} \\
 & & & \\
\forall s \in \{s_a, s_b\}\ \forall x \in \{a, b\}\ (\delta(s, x) & = & (s, L)) & \text{erstes Zeichen suchen} \\
 & & & \\
\delta(s_a, \#) & = & (s'_a, R) & \text{zum ersten Zeichen gehen} \\
\delta(s_b, \#) & = & (s'_b, R) & \\
 & & & \\
\delta(s'_a, a) & = & (s_Y, R) & \text{auf Antwort vorbereiten} \\
\delta(s'_a, b) & = & (s_N, R) & \\
\delta(s'_b, a) & = & (s_N, R) & \\
\delta(s'_b, b) & = & (s_Y, R) & \\
 & & & \\
\forall s \in \{s_Y, s_N\}\ \forall x \in \{a, b\}\ (\delta(s, x) & = & (s, R)) & \text{rechtes Ende suchen} \\
 & & & \\
\delta(s_Y, \#) & = & (s_f, 1) & \text{Antwort schreiben} \\
\delta(s_N, \#) & = & (s_f, 0) & \\
 & & & \\
\forall x \in \{0, 1\}\ (\delta(s_f, x) & = & (h, R)) & \text{Endposition einnehmen}
\end{array}
$$

Alle nicht aufgeführten Übergänge sind undefiniert.